Computer Science

AN OBJECT-ORIENTED TYPE SYSTEM FOR COMPUTATIONAL ALGEBRA

Christina M. Knapp Dr. Robert F. Morse*
University of Evansville
1800 Lincoln Avenue
Evansville, IN 47722
ck38@evansville.edu

The GAP (Groups, Algorithms, Programming) system for computational discrete mathematics was originally specialized in group theory, but developments in group theoretic algorithms required the calculation with other mathematical objects, such as vectors, algebras and polynomials. Computational support for semigroups have also been introduced into the GAP library which in turn requires its own computational infrastructure including groups, finite automata and lattices. The purpose of this presentation is to describe the GAP 4 type system and its use in the design and implementation of lattices in the GAP.

A major design goal of GAP 4 is extensibility so that without rewriting any existing code, new algebraic objects can be added to the system. This is accomplished by the GAP 4 type system and it methods selection scheme. In the usual object-oriented approach, methods applicable to an object are actually or implicitly stored as part of the object. Objects are gathered in classes and the class of the object actually determines the methods applicable to it. This setup has two major limitations relative to the requirement of extensibility and the application area of algebraic computation.

First, identical methods can be associated with more than one object and there is no natural way of deciding which one should supply the method. For lattices, multiplication (the meet operation) was implemented for lattice elements. However, lattice elements themselves may be objects which one may wish to multiply (on both sides) by say rational numbers. In the normal set up this would inevitably require changes to the existing rational number class. It is clearly undesirable for an extension to the system to require such modifications to the existing system. GAP overcomes this by allowing method selection based on types of all arguments and on certain aspects of the relationship between arguments. Methods are separate from the objects and allows for definition of methods on other attributes other then type.

Second, in algebraic computations one defines objects in a highly implicit manner. We are given a set of generators for a lattice and nothing else. As one works with this object we find out information that can make subsequent calculations more efficient. For example, a lattice is determined to be a chain then finding the complement of any element is very simple. Hence the GAP 4 type system allows for the type of an object to change over time to optimize algorithm efficiency.

*This research was sponsored by an ADVANTAGE Undergraduate Research Grant from the University of Evansville.